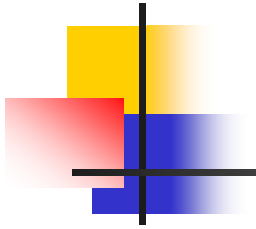
Challenge: Platform-dependent Adaptation of Irregular Applications

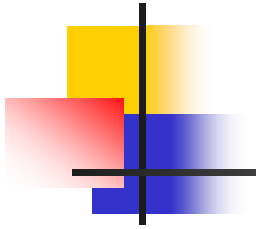
Kai Shen

University of Rochester



Overview

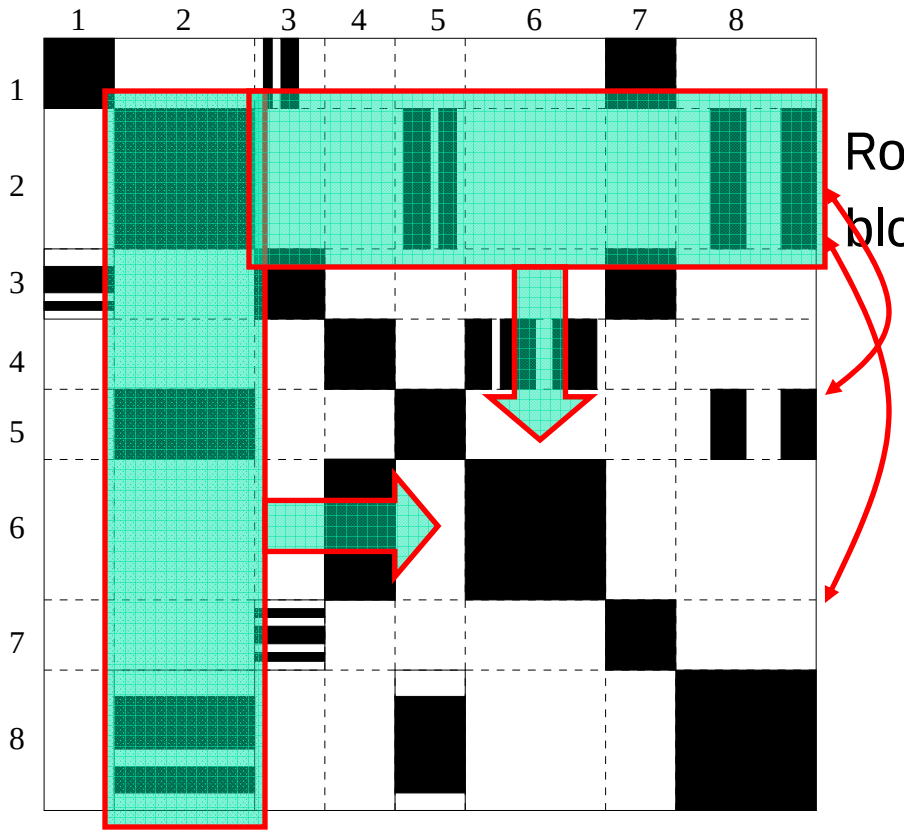
- Large variation on platform characteristics
  - High-end machine: 1us latency, 1000MB/sec throughput
  - Low-end PC cluster: 100us latency, 50MB/sec throughput
- Performance portability is desirable
  - If not for end applications, more certain for computational libraries
- Application adaptation
  - Particularly effective when tradeoffs can be made among multiple metrics (comm. frequency/volume, computation)
  - Difficult for irregular applications



Parallel Sparse LU Factorization

- LU factorization with partial pivoting: used for solving a linear system $Ax = b$ ($PA=LU$).
- Applications:
  - Device/circuit simulation, fluid dynamics, … …
  - In the Newton's method for solving non-linear systems
- Characteristics for parallel sparse LU factorization:
  - Runtime data structure variation
  - Non-uniform computation/communication patterns
  - Execution patterns dependent on input numeric values

Data Structure and Computation Steps in the Supernodal Approach



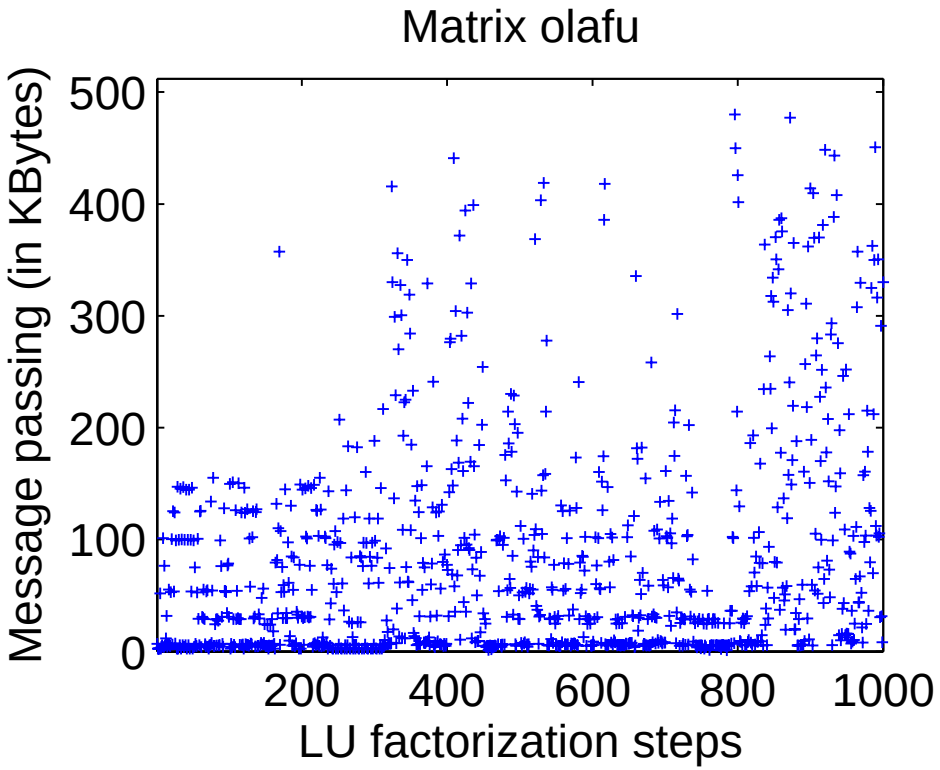

```
for each column block K (1→N)
    Perform Factor(K);
    Perform SwapScale(K);
    Perform Update(K);
endfor
```

Processor mapping:

- 1-D cyclic
- 2-D cyclic (more scalable)

Application Characteristics: Irregular Execution Patterns

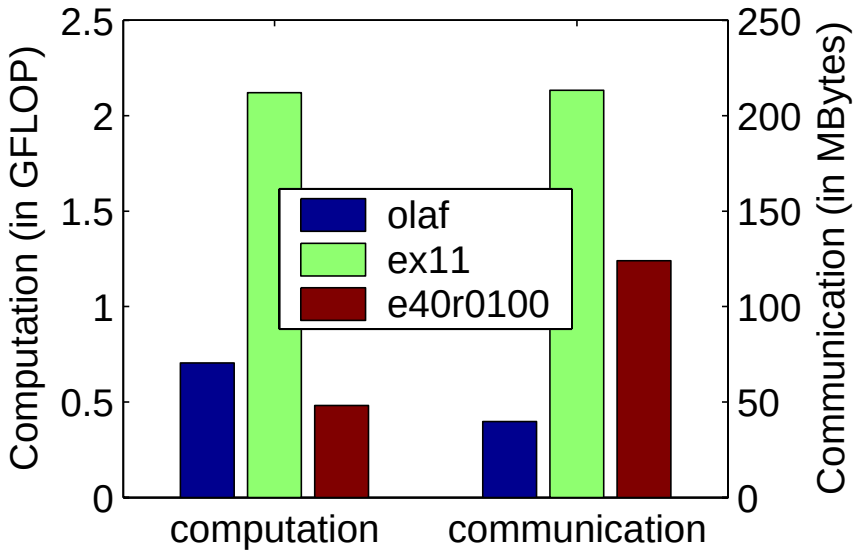
The comm. volume at process 0 for the first 1000 steps of factorization, using the S+ solver [shen et al., 2000]



Application Characteristics: Value-dependent Execution Patterns

- The application computation/communication pattern is dependent on input numerical values

Per-process computation/communication volume for three matrices of similar sizes



Platform-dependent Application Adaptation

- Techniques exist to exploit tradeoffs among multiple metrics (communication, computation, num. stability)
- Challenging to predict such tradeoffs quantitatively with specific input on particular parallel computing platforms

- Possible approaches:
  - Performance model-driven adaptation
  - Sampling-driven adaptation
  - Phase-based adaptation

- Need assistance from software tools, programming environments, and runtime systems